ESTRUCTURA DE DATOS PARA LA REPRESENTACIÓN EFICIENTE DE ARCHIVOS Y DIRECTORIOS EN UN SISTEMA DE ARCHIVOS.

Jamerson Stive Correa Correa
Universidad EAFIT
Colombia
jscorreac@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

1. INTRODUCCIÓN

Gracias a que nos encontramos actualmente en la era de la Información, se hace imprescindible para la humanidad desarrollar herramientas informáticas que le permitan tener acceso inmediato a volúmenes grandes de información, es decir, dichas herramientas deben estar en capacidad de organizar la información y permitir al ser humano tener un acercamiento eficaz y eficiente a la misma.

Este es el caso de los Sistemas de archivos los cuales son un componente de un sistema operativo encargado de asignar espacio a los archivos, administrar el espacio libre y permitir el acceso a los datos guardados. [1]

2. PROBLEMA

A través del análisis de algoritmos y estructuras de datos, lograr elaborar e implementar un Sistema de archivos que permita a los usuarios consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio. Esto con el fin de lograr optimizar y suplir la necesidad de almacenar y procesar grandes volumenes de información.

3. TRABAJOS RELACIONADOS

3.1 Árból AA

El Árbol AA es denomidado así por las iniciales de su creador, Arne Andersson.

Estos árboles son una variación del Árbol rojo-negro, se caracterizan por generar un menor número de situaciones para ser analizadas, es decir, se encargan de simplificar los ordenamientos a realizar para llevar a cabo las operaciones de balanceo del árbol. A diferencia del Árbol rojo-negro, el cual debe examinar siete situaciones, éste solo debe tener en cuenta dos. Todo esto gracias a que, además de tener en cuenta las reglas que rigen a los árboles coloreados, dicho árbol solo permite como hijo derecho a los nodos rojos. [2]

Las dos operaciones que realiza el Árbol AA son la torsión, llevada a cabo cuando una inserción o un borrado generan un enlace horizontal izquierdo, y la división, que tiene lugar cuando una inserción o un borrado generan dos enlaces horizontales derechos. [3]

3.2 Árbol de segmento

En ciencia de la computación, un **árbol de segmento** (en inglés: *Segment tree*) es una estructura de datos en forma de árbol para guardar intervalos o segmentos. Permite consultar cuál de los segmentos guardados contiene un punto. Este es, en principio, una estructura estática; es decir, su contenido no puede ser modificado una vez que su estructura es construida. Algunas aplicaciones del árbol de segmento son vistas en las áreas de la geometría computacional y en los sistemas de información geográfica.[4]

3.3 Árbol B*

Una variante del árbol B es el árbol B*, propuesta por Knuth en 1973, cuya peculiaridad consiste en que se aumenta el número mínimo de valores de clave que puede contener un nodo que no sea la raíz, de forma que en lugar de garantizar un 50% de utilización de espacio como en el árbol B, se aproveche como mínimo las dos terceras partes del mismo [5].

Para realizar los procedimientos de árboles B^* también se debe abordar la cuestión de dividir la raíz, la cual, por definición, nunca tiene hermanos. Si no existen hermanos, no es posible la división de dos a tres. Knuth sugiere permitir que la raíz crezca hasta un tamaño mayor que los demás nodos, de tal forma que, cuando se divida, pueda producir dos nodos cada uno lleno casi a las dos terceras partes. Esta sugerencia tiene la ventaja de asegurar que todos los nodos por debajo del nivel de la raíz se adhieren a las características de los árboles B^*. Sin embargo, tiene la desventaja de requerir que los procedimientos sean capaces de manejar un nodo que sea de mayor tamaño que todos los demás.[6]

3.4 Tablas de hash

Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información.

La idea surge de los arreglos que nos permiten acceso a sus elementos en orden 0(1). Una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos almacenados a partir de una clave de este. [7]

Una tabla *hash* tiene como principal ventaja que el acceso a los datos suele ser muy rápido si se cumplen las siguientes condiciones:

- Una razón de ocupación no muy elevada (a partir del 75% de ocupación se producen demasiadas colisiones y la tabla se vuelve ineficiente).

- Una función resumen que distribuya uniformemente las claves. Si la función está mal diseñada, se producirán muchas colisiones. [8]

REFERENCIAS

1. *Alberto, R., Mauricio, T. Estructura de datos para listar el contenido de un directorio. Universidad EAFIT escuela de ingeniería, departamento de informática y sistemas. file:///C:/Users/estiv/AppData/Local/Temp/Rar$DIa0.021/Proyecto%20Final%20ED1%20Sistema-directorios%20Vr%203.0.pdf*

2. *Estructura de datos-Árbol AA.(2015)* https://es.wikipedia.org/wiki/%C3%81rbol_AA

3. *Leopoldo, S.B., Capítulo 15, Árboles AA. Universidad Técnica Federico Santa María.(2010).* http://www2.elo.utfsm.cl/~lsb/elo320/clases/c15.pdf

4. *Estructura de datos-Árbol de segmento. (2016).* https://es.wikipedia.org/wiki/%C3%81rbol_de_segmento

5. *Juan, B.V., Árboles B*. Universidad Catolica de Oriente. (2011).* https://sites.google.com/site/tutoriasarboles/arbolesb

6. *Daniel, L., Abraham, G.S., Martin, G., Antonio Jose Director de proyecto: Joaquín, F.V., Árbol B*Universidad de Granada,* http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B2.htm

7. *Edgardo, F.M., Tablas hash, Instituto Politécnico Nacional, Escuela superior de Cómputo,* http://www.eafranco.com/docencia/estructurasdedatos/files/05/Tema05.pdf

8. *Estructuras de datos-Tablas de hash. (2017). https://es.wikipedia.org/wiki/Tabla_hash#Ventajas_e_inconvenientes_de_las_tablas_hash*